これで、認証がすべて完了したので、ユーザーは登録、ログイン、ログアウトができます。

これらのユーザーインターフェイスはすべて適切に設定されています。

実際にチャット画面に移動し、ここでテキストフィールドに入力したメッセージの保存を開始します。そのため、彼らがそのメッセージを送信した瞬間は、Firebase Cloud Fire herestoreに直接送信され、データベース内で確認できるようになります。

いつものように、Flutterに関してはドキュメントが不足することはなく、Firebaseでも同じことが言えます。

両方のFlutterパッケージページには多くのドキュメントがありますので、ここで紹介しているDartコードになります。

しかし、Firebaseに移動してドキュメントに移動すると、すべてのプラットフォーム、iOS、Android、Web向けの大量のドキュメントを見つけることができます。本当にすぐに。

両方のチームがそれに一生懸命取り組んでいるに違いない。データをクラウドに保存するために、Firestoreを使用するには、まずFirebase Webサイトでいくつかの設定を行う必要があります。

そのため、ここでデータベースタブに移動する必要があります。次に、Create databaseをクリックして、Cloud Firestoreを作成できるようにします。

Firebaseが提供する別の種類のデータベースであるリアルタイムデータベースがあります。

今、これは彼らの一種のデータベースだと思います。

非常に高速ですが、JSONの保存に基づいています。JSONは、NoSQLデータベースに基づくFirestoreほどの能力や拡張性はありません。 Cloud Firestoreである最新かつ最高のものを使用します。

[データベースの作成]をクリックして、テストモードで開始するように切り替えて、データベース参照を持つすべてのユーザーがデータベースの読み取りと書き込みを行えるようにします。テストが完了し、アプリストアにアプリを展開する準備が整ったら、これらのルールを少し安全に変更します。

しかし今のところ、私たちのアプリについてはまだ誰も知りませんし、気にしません。

それでは、先に進んで[有効]をクリックしてCloud Firestoreを作成しましょう。

セットアップが完了すると、データベースダッシュボードが次のように表示されます。

そして、ここにコレクションを追加する機能があります。

それでは、コレクションとは何ですか？

ショップを経営していると想像してみてください。また、在庫を管理する必要があり、顧客を管理する必要があります。そのため、顧客向けにExcelスプレッドシートを作成できます。

あなたはあなたの製品のためにそれを持っているかもしれません。

サプライヤ用に1つ持っているかもしれません。

そして、これらのExcelスプレッドシートはコレクションと同等です。

したがって、messagesという名前のメッセージを作成する必要があります。これには、すべてのメッセージと、それらのメッセージを送信した送信者も保存されます。そして、いくつかのフィールドを作成できるように、次をクリックします。

そこで、ここでサンプルデータを作成して、いくつかのフィールドを確立します。

したがって、最初のフィールドはtextという名前になります。したがって、これはメッセージテキストになります。

文字列になります。

そして、それにダミー値を与えましょう。

こんにちはと言ってみましょう。

そして、送信者フィールドになる別のフィールドを追加する準備が整いました。

もう一度別の文字列で、最初の送信者がangela@email.comであったとしましょう。

保存してみましょう。最初のコレクションと最初のデータがデータベース内に表示されます。そこにあります。

Firebaseダッシュボードからデータを追加するのは非常に簡単です。

しかし、ユーザーが入力している内容に基づいて、アプリからこのデータを追加できるようにする必要があります。そのためには、chat\_screen.dartに移動し、Cloud Firestoreパッケージを入力する必要があります。

そして、ユーザーが変数に入力したメッセージを保存することができます。多分それをmessage Textと呼びましょう。ここで、テキストフィールドの値をそのmessageText変数に設定します。

そのため、値に等しい値を設定します。

これで、loggedInUser.emailとともにメッセージテキストを使用して、そのデータをCloud Firestoreに送信できるようになりました。そして、ユーザーが「送信」ボタンを押したときにそれを実行します。そのため、ここでそのonPressedコールバック内で実行します。を介してmessageTextにアクセスできることを知っています

このプロパティはmessageTextであり、loggedInUser.emailである送信者にもアクセスできます。

また、コレクションと呼んだもの、フィールド、送信者、テキストと呼んだものを思い出すことも非常に重要です。

大文字と小文字を含めてまったく同じように入力したコードで使用する場合は、必ず一致させる必要があります。

これで、Cloud Firestoreの新しいインスタンスを作成する準備が整いました。

したがって、authを使用して行ったように、Firestore.instanceに相当するFirestoreを作成し、ここでonPressedでFirestoreを使用して、選択したコレクションを利用できます。

これで、パスはコレクションの名前に指定した文字列になります。

ですから、私の場合はメッセージでなければなりません。そのため、ここにメッセージを入れます。

Firebaseコンソールにあるものとまったく同じであることを確認してください。

これで、データを追加する準備が整いました。また、addメソッドはマップデータタイプを想定していることに注意してください。

したがって、キーとして文字列を持つマップである必要があり、値は任意のデータ型にすることができます。

中括弧を追加してマップを作成し、キーをコレクション内のドキュメントのフィールドとして指定したものと一致させる必要があります。

送信者とテキストがあります。

最初のキーとしてテキストから始めましょう。もちろん、その値は、ユーザーがテキストフィールドに入力したmessageTextである必要があります。

そして、2番目は送信者になり、送信者フィールドにはログインしているユーザーの電子メールアドレスが入ります。

これは、送信したメッセージを確認するためのユーザーを識別する最も簡単な方法です。

これで、テストする準備が整いました。

それでは、実行して、アプリをチェックしてみましょう。先に進み、angela @ email.comとしてログインしてパスワードを入力し、[ログイン]をクリックして、メッセージを入力します。

「Hello World」と入力して、送信をクリックします。

まだ処理していないので、ここでは何も起こりませんが、コンソールにアクセスして更新を押すか、ここの小さなボタンをクリックすると、コンソールが読み込まれると、メッセージのコレクションが表示されます現在は2つのデータです。

これは元のものであり、これはアプリを介して今追加したものです。

これで、Cloud Firestoreのコレクションにデータを保存できるようになりました。

そして、データを取得するだけです。

これが、今後のレッスンで行うことです。